Accelerating Best Response Calculation in Large Extensive Games
Suppose you have a 2-player game.

You can use an algorithm for learning a strategy in this space.
There are several ways to evaluate a strategy.

You could run a competition against other agents.

- Computer Poker Competition
- RoboCup
- Computer Olympiad
- Trading Agent Competition

Wednesday, November 14, 2012
Worst-case performance is another useful metric.

2-player games:
Use Expectimax to find a best-response counterstrategy.
Optimal Strategy (2-player, zero-sum game):

Nash Equilibrium, maximize worst-case performance.

Or, equivalently, minimize worst-case loss.
Compete against other agents?  
Worst case performance?

2-player Limit Texas Hold’em Poker:

~$10^{18}$ game states  
~$10^{14}$ information sets  
(design decision points)

Computing an optimal strategy:  
4 PB of RAM,  
1400 cpu-years
2-player Limit Texas Hold’em Poker:

~$10^{18}$ game states

Computing a best response:
Thought to be intractable, may require a full game tree traversal.

At 3 billion states/sec, would take 10 years.
State-space abstraction allows us to produce strategies for the game.
Evaluation has relied on tournaments.

AAAI 2007 - Phil Laak
First Man-Machine Poker Championship

Annual Computer Poker Competition:
2006-2011
(at AAAI next month!)
Key questions:
How much did abstraction hurt?  
How good are the agents, really?

Can we make the best-response computation tractable to find out?
Accelerating Best Response Computation

Four ways to speed up best response calculation in imperfect information games

Formerly intractable computations are now run in one day

Solving an 8 year old evaluation problem

How good are state-of-the-art computer poker programs?
Expectimax Search

The Best Response Task:

Given an opponent’s entire strategy, choose actions to maximize our expected value.
Our View  Cards are Private  Opponent’s View

Wednesday, November 14, 2012
Our View

Opponent's View

Opponent Choice
Nodes (probabilities are known)
To determine our payoff here...
Our Tree

Opponent’s Tree

..we need to compute the distribution over these states
Expectimax Search

Simple recursive tree walk:

❤️ Pass forward:
   Probability of opponent being in their private states

♣ Return:
   Expected value for our private state
Expectimax Search

Simple recursive tree walk:

Pass forward:
- Probability of opponent being in their private states

Return:
- Expected value for our private state

Visits each state just once!
- But $10^{18}$ states is still intractable.
Accelerated Best Response

Four ways to accelerate this computation:

1) Take advantage of what the opponent doesn’t know

2) Do $O(n^2)$ work in $O(n)$ time

3) Avoid isomorphic game states

4) Parallel computation
What the opponent doesn’t know

My Tree

Your Tree
What the opponent doesn’t know

My Tree

Your Tree
What the opponent doesn’t know

My Tree

Your Tree
We can instead walk this much smaller tree of public information.

At each node, we choose actions for all of the states our opponent cannot tell apart.

More work per node, but we reuse queries to the opponent’s strategy!

\[ \sim \text{110x speedup in Texas hold’em} \]
Accelerated Best Response

The new technique has four orthogonal improvements:

1) Take advantage of what the opponent doesn’t know

2) Do $O(n^2)$ work in $O(n)$ time.

3) Avoid isomorphic game states

4) Parallel computation
Fast Terminal Node Evaluation
Fast Terminal Node Evaluation

Opponent’s $n$ States

My $n$ States
Fast Terminal Node Evaluation

Opponent's $n$ States

My $n$ States

\[ p_1 u_1 + p_2 u_2 + p_3 u_3 + p_4 u_4 + p_5 u_5 + p_6 u_6 \]

\[ = p_1 u_1 + p_2 u_2 + p_3 u_3 + p_4 u_4 + p_5 u_5 + p_6 u_6 \]

\[ = p_1 u_1 + p_2 u_2 + p_3 u_3 + p_4 u_4 + p_5 u_5 + p_6 u_6 \]

\[ = p_1 u_1 + p_2 u_2 + p_3 u_3 + p_4 u_4 + p_5 u_5 + p_6 u_6 \]

\[ = p_1 u_1 + p_2 u_2 + p_3 u_3 + p_4 u_4 + p_5 u_5 + p_6 u_6 \]

\[ = p_1 u_1 + p_2 u_2 + p_3 u_3 + p_4 u_4 + p_5 u_5 + p_6 u_6 \]

Wednesday, November 14, 2012
Fast Terminal Node Evaluation

Opponent’s $n$ States

My $n$ States

$O(n^2)$ work to evaluate $n$ hands
Most games have structure that can be exploited. In Poker, states are ranked, and the highest rank wins.
To calculate one state’s EV, we only need:
- Probability of opponent reaching weaker states
- Probability of opponent reaching stronger states

\[ \text{EV}[i] = p(\text{lose}) \times \text{util(lose)} + p(\text{win}) \times \text{util(win)} \]
By exploiting the game’s structure, we can use two for() loops instead of two nested for() loops.

$O(n^2)$ to $O(n)$. 7.7x speedup in Texas hold’em.

(Some tricky details resolved in the paper)
Accelerated Best Response

The new technique has four orthogonal improvements:

1) Take advantage of what the opponent doesn’t know

2) Do $O(n^2)$ work in $O(n)$ time.

3) Avoid isomorphic game states

4) Parallel computation
Avoid Isomorphic States

21.5x reduction in game size

(only correct if opponent’s strategy also does this)
Accelerated Best Response

The new technique has four orthogonal improvements:

1) Take advantage of what the opponent doesn’t know, to walk the much smaller public tree

2) Use a fast terminal node evaluation to do O(n^2) work in O(n) time.

3) Avoid isomorphic game states

4) Parallel computation
Parallel Computation

- 24,570 equal sized independent subtrees.
- Takes 4m30s to solve each one.
- $24,570 \times 4.5 \text{ minutes} = 76 \text{ cpu-days}$
Parallel Computation

♥ 24,570 equal sized independent subtrees.

♣ Takes 4m30s to solve each one.

♦ 24,570 * 4.5 minutes = 76 cpu-days

♠ 72 processors on a cluster: 1 day computation!
Evaluating the Progress of Computer Poker Research
Evaluating Computer Poker Agents

♥  Annual Computer Poker Competition (ACPC)
- Started in 2006
- Hosted at AAAI this year
- 2-player Limit: Strongest agents are competitive with world’s best human pros

♣  Most successful approach
   (U of A, CMU, many others):
   - Approximate a Nash equilibrium,
     worst case loss of $0 per game

♦  For the first time, we can now tell how close we are to this goal!
### Trivial Opponents

<table>
<thead>
<tr>
<th>Opponent Style</th>
<th>Value for Best Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always-Fold</td>
<td>750</td>
</tr>
<tr>
<td>Always-Call</td>
<td>1163.48</td>
</tr>
<tr>
<td>Always-Raise</td>
<td>3697.69</td>
</tr>
<tr>
<td>Uniform Random</td>
<td>3466.32</td>
</tr>
</tbody>
</table>

A human professional’s goal is to win 50.
An optimal strategy would lose 0.
(Units are milli-big-blinds per game)
University of Alberta Agents

- Computer Poker Competition
- Man-vs-Machine

<table>
<thead>
<tr>
<th>Year</th>
<th>Best Response (mbb/g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006</td>
<td>320</td>
</tr>
<tr>
<td>2007</td>
<td>240</td>
</tr>
<tr>
<td>2008</td>
<td>160</td>
</tr>
<tr>
<td>2009</td>
<td>80</td>
</tr>
<tr>
<td>2010</td>
<td>0</td>
</tr>
<tr>
<td>2011</td>
<td>0</td>
</tr>
</tbody>
</table>

Wednesday, November 14, 2012
University of Alberta Agents

2007 Man-Machine: Narrow Human Win

Best Response (mbb/g)

Year

2006 2007 2008 2009 2010 2011
University of Alberta Agents

2007 Man-Machine: Narrow Human Win

2008 Man-Machine: Narrow Computer Win
Evaluating the University of Alberta agents

Comparing Abstraction Techniques:

- Percentile HS
- Public PHS
- k-Means Earthmover

Best Response (mbb/g) vs. Abstraction Size (# information sets)
Evaluating Computer Poker Agents: 2010 Competition

<table>
<thead>
<tr>
<th></th>
<th>Rock hopper</th>
<th>GGValuta</th>
<th>HyperB (UofA)</th>
<th>PULPO</th>
<th>GS6 (CMU)</th>
<th>Littlerock</th>
<th>Best Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rock hopper</td>
<td>6</td>
<td>3</td>
<td>7</td>
<td>37</td>
<td>77</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>GGValuta</td>
<td>-6</td>
<td>-6</td>
<td>3</td>
<td>1</td>
<td>31</td>
<td>77</td>
<td>237</td>
</tr>
<tr>
<td>HyperB (UofA)</td>
<td>-3</td>
<td>-3</td>
<td>2</td>
<td>31</td>
<td>70</td>
<td>135</td>
<td></td>
</tr>
<tr>
<td>PULPO</td>
<td>-7</td>
<td>-1</td>
<td>-2</td>
<td>32</td>
<td>125</td>
<td>399</td>
<td></td>
</tr>
<tr>
<td>GS6 (CMU)</td>
<td>-37</td>
<td>-31</td>
<td>-31</td>
<td>-32</td>
<td>47</td>
<td>318</td>
<td></td>
</tr>
<tr>
<td>Littlerock</td>
<td>-77</td>
<td>-77</td>
<td>-70</td>
<td>-125</td>
<td>-47</td>
<td>421</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

✿ Fast best-response calculation in imperfect information games

♦ The previously intractable computation can now be run in a day!

♣ Computer poker community is making steady progress towards robust strategies

♠ Many additional exciting results in the paper and at the poster!
More details at our poster!
Today, 4:00 - 5:20, Room 120-121
Additional Slides:

- Expectimax
- Abstraction
- Public Tree
- Pathologies
- n^2 to n
- CFR
- Polaris
- Tilting
- Hyperborean 2009
- Additional Graphs

Wednesday, November 14, 2012
# Leduc Hold’em Pathologies

<table>
<thead>
<tr>
<th>Abstraction</th>
<th>Best Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Game vs Real Game</td>
<td>0</td>
</tr>
<tr>
<td>J.Q.K vs Real Game</td>
<td>55.2</td>
</tr>
<tr>
<td>[JQ].K vs Real Game</td>
<td>69.0</td>
</tr>
<tr>
<td>J.[QK] vs Real Game</td>
<td>126.3</td>
</tr>
<tr>
<td>[JQK] vs Real Game</td>
<td>219.3</td>
</tr>
<tr>
<td>[JQ].K vs [JQ].K</td>
<td>272.2</td>
</tr>
<tr>
<td>[JQ].K vs J.Q.K</td>
<td>274.1</td>
</tr>
<tr>
<td>Real Game vs J.[QK]</td>
<td>345.7</td>
</tr>
<tr>
<td>Real Game vs [JQ].K</td>
<td>348.9</td>
</tr>
<tr>
<td>J.Q.K vs J.Q.K</td>
<td>359.9</td>
</tr>
<tr>
<td>J.Q.K vs [JQ].K</td>
<td>401.3</td>
</tr>
<tr>
<td>J.[QK] vs J.[QK]</td>
<td>440.6</td>
</tr>
<tr>
<td>Real Game vs [JQK]</td>
<td>459.5</td>
</tr>
<tr>
<td>Real Game vs J.Q.K</td>
<td>491.0</td>
</tr>
<tr>
<td>[JQK] vs [JQK]</td>
<td>755.8</td>
</tr>
</tbody>
</table>
Expectimax

My Tree

Your Tree

Reach:
2: 0.5
K: 0.5
Conventional Best Response in one tree walk

My Tree

Your Tree

Reach:
2: 0.5*0.75
K: 0.5*0.1

-0.29

0.38

0.05

Home
Conventional Best Response in one tree walk

My Tree

Your Tree

Reach:
2: 0.5*0.75
K: 0.5*0.1
Conventional Best Response in one tree walk

My Tree

Your Tree

Reach:
2: 0.5*0.75*0.25
K: 0.5*0.1*0.9
Conventional Best Response in one tree walk

My Tree

Your Tree

Reach:
2: 0.5*0.75
K: 0.5*0.1

-0.29
0.38
0.05

-0.045

Wednesday, November 14, 2012
Conventional Best Response in one tree walk

My Tree

Your Tree

Reach:
2: 0.5*0.75*0.75
K: 0.5*0.1*0.1
Conventional Best Response in one tree walk

My Tree

Your Tree

Reach:
2: 0.5*0.75
K: 0.5*0.1

-0.29

0.1

0.38

0.05
Conventional Best Response in one tree walk

My Tree

Your Tree

Reach:
2: 0.5*0.75
K: 0.5*0.1

Home
Conventional Best Response in one tree walk

My Tree

Your Tree

Reach:
2: 0.5*0.75
K: 0.5*0.1

Wednesday, November 14, 2012
Conventional Best Response in one tree walk

My Tree

Your Tree

Reach:
2: 0.5*0.75
K: 0.5*0.1
Conventional Best Response in one tree walk

My Tree

Your Tree

Reach:
2: 0.5
K: 0.5

Home
Conventional Best Response in one tree walk

My Tree

Your Tree

Reach:
2: 0.5
K: 0.5
Conventional Best Response in one tree walk

My Tree

Your Tree

Reach:
2: 0.5
K: 0.5

Wednesday, November 14, 2012
Conventional Best Response in one tree walk

My Tree

Your Tree

Reach:
2: 0.5*0.25
K: 0.5*0.9
I: Walking the Public Tree

Their Reach Prob:
2: 0.5
K: 0.5

My Value:
2:
K:
Walking the Public Tree

Their Reach Prob:
2: 0.5*0.25
K: 0.5*0.9

My Value:
2:
K:
I: Walking the Public Tree

Their Reach Prob:
2: 0.5*0.25
K: 0.5*0.9

My Value:
2: -0.45
K: 0.13

-0.45, 0.13
I: Walking the Public Tree

Their Reach Prob:
- 2: 0.5*0.25
- K: 0.5*0.9

My Value:
- 2: -0.45
- K: 0.13
I: Walking the Public Tree

Their Reach Prob:
2: 0.5*0.25
K: 0.5*0.9

My Value:
2: -0.29
K: -0.29
I: Walking the Public Tree

Their Reach Prob:
2: 0.5*0.25
K: 0.5*0.9

My Value:
2: -0.29
K: 0.13
I: Walking the Public Tree

Their Reach Prob:
2: 0.5
K: 0.5

My Value:
2: -0.29
K: 0.13

-0.29, 0.13
I: Walking the Public Tree

Their Reach Prob:
2: 0.5*0.75
K: 0.5*0.1

My Value:
2:
K:
I: Walking the Public Tree

Their Reach Prob:
2: 0.5*0.75
K: 0.5*0.1

My Value:
2:
K:

-0.29, 0.13

-0.29
0.13

1: Walking the Public Tree

Wednesday, November 14, 2012
I: Walking the Public Tree

Their Reach Prob:
2: 0.5 * 0.75
K: 0.5 * 0.1

My Value:
2: -0.05
K: 0.09

-0.29, 0.13

-0.05, 0.09

0.13

-0.29

-0.05

0.09

Home
I: Walking the Public Tree

Their Reach Prob:
2: 0.5*0.75
K: 0.5*0.1

My Value:
2: -0.05
K: 0.09

-0.29, 0.13

-0.05, 0.09
I: Walking the Public Tree

Their Reach Prob:
2: 0.5*0.75
K: 0.5*0.1

My Value:
2: 0.14
K: 0.14

-0.29, 0.13

-0.05, 0.09
0.14

-0.05
0.14
0.09
0.14

-0.29
0.13

Home

Wednesday, November 14, 2012
I: Walking the Public Tree

Their Reach Prob:
2: 0.5*0.75
K: 0.5*0.1

My Value:
2: 0.14
K: 0.14
I: Walking the Public Tree

Their Reach Prob:
2: 0.5*0.75
K: 0.5*0.1

My Value:
2: 0.14
K: 0.14

-0.29, 0.13

0.09, 0.23
I: Walking the Public Tree

Their Reach Prob:
2: 0.5*0.75
K: 0.5*0.1

My Value:
2: -0.05
K: 0.19

-0.29, 0.13

-0.29
0.13

0.09
0.23

0.09
0.23

-0.05

0.13
0.23

-0.05

0.19

0.19
I: Walking the Public Tree

Their Reach Prob:
2: 0.5*0.75
K: 0.5*0.1

My Value:
2: 0.09
K: 0.23

-0.29, 0.13

0.09 0.23
0.09
0.23

-0.05 0.19

-0.29
0.09 0.13
0.23

Home
I: Walking the Public Tree

Their Reach Prob:

2: 0.5*0.75
K: 0.5*0.1

My Value:

2: -0.2
K: 0.36
I: Walking the Public Tree

Their Reach Prob:
2: 0.5*0.75
K: 0.5*0.1

My Value:
0.18
<table>
<thead>
<tr>
<th>Agent</th>
<th>Size</th>
<th>Tilt</th>
<th>Best Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pink</td>
<td>266m</td>
<td>0, 0, 0, 0</td>
<td>235.294</td>
</tr>
<tr>
<td>Orange</td>
<td>266m</td>
<td>7, 0, 0, 7</td>
<td>227.457</td>
</tr>
<tr>
<td>Peach</td>
<td>266m</td>
<td>0, 0, 0, 7</td>
<td>228.325</td>
</tr>
<tr>
<td>Red</td>
<td>115m</td>
<td>0, -7, 0, 0</td>
<td>257.231</td>
</tr>
<tr>
<td>Green</td>
<td>115m</td>
<td>0, -7, 0, -7</td>
<td>263.702</td>
</tr>
<tr>
<td>(Reference)</td>
<td>115m</td>
<td>0, 0, 0, 0</td>
<td>266.797</td>
</tr>
</tbody>
</table>
Polaris Hyperborean

Man-vs-Machine 2007
Narrow loss
Polaris

Man-vs-Machine 2007
Narrow loss

Man-vs-Machine 2008
Narrow win
Tilting

Exploitability (mb/g)

Percent bonus for winner

Wednesday, November 14, 2012
Counterfactual Regret Minimization: Abstract-Game Best Response

10-bucket Perfect Recall, Percentile 10 $E[H_2^2]$

Exploitability (mbb/g)

Iterations (million)
Counterfactual Regret Minimization: Real Game Best Response

10-bucket Perfect Recall, Percentile 10 $E[H_S^2]$
Hyperborean 2009

Polaris

Best Response (mbb/g)

Year

2006 2007 2008 2009 2010 2011?

Wednesday, November 14, 2012
Abstraction: Perc HS$^2$
Abstraction: k-Means
Abstraction: HS Distributions

Distribution over future outcomes for hand AsAd
Abstraction: HS Distributions

Distribution over future outcomes for hand 2s7c

E[HS]
k-Means Earthmover Abstraction

Exploitability of KE/KO families

Abstraction size (in millions of infosets)

Exploitability (mb/s/g)

KE/KO

Flop KE/KO

20-pub KE/KO

Flop 20-pub KE/KO

Perfect flop

18630

3700

9000

34460

20x3000

100k-34k

20x9000

20x9000
3: Fast Terminal Node Evaluation

His Reach Probs:  

<table>
<thead>
<tr>
<th>Card</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ace</td>
<td>0.1</td>
</tr>
<tr>
<td>King</td>
<td>0.05</td>
</tr>
<tr>
<td>9</td>
<td>0.02</td>
</tr>
</tbody>
</table>

My Values:  

<table>
<thead>
<tr>
<th>Card</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ace</td>
<td>?</td>
</tr>
<tr>
<td>King</td>
<td>?</td>
</tr>
<tr>
<td>9</td>
<td>?</td>
</tr>
</tbody>
</table>
3: Fast Terminal Node Evaluation

His Reach Probs:  

My Values:  

\[u = \text{utility for winner}\]

\[= 0 \times 0.1 + u \times 0.05 + u \times 0.02 + \ldots\]

\[= -u \times 0.1 + 0 \times 0.05 + u \times 0.02 + \ldots\]

\[= -u \times 0.1 + -u \times 0.05 + 0 \times 0.02 + \ldots\]

\[\ldots\]
3: Fast Terminal Node Evaluation

The obvious $O(n^2)$ algorithm:

\[
\begin{align*}
  r[i] &= \text{his reach probs} \\
  v[i] &= \text{my values} \\
  u &= \text{utility for the winner}
\end{align*}
\]

\[
\begin{align*}
  &\text{for( } a = \text{each of my hands } ) \\
  &\quad \text{for( } b = \text{each of his hands } ) \\
  &\quad \quad \text{if( } a > b \text{ )} \\
  &\quad \quad \quad v[a] += u \times r[b] \\
  &\quad \quad \text{else if( } a < b \text{ )} \\
  &\quad \quad \quad v[a] -= u \times r[b]
\end{align*}
\]
But games are fun because they have structure in determining the payoffs, and we can take advantage of that.

This Vector-vs-Vector evaluation can often be done in $O(n)$ time, and not just in poker.
### 3: Fast Terminal Node Evaluation

<table>
<thead>
<tr>
<th>Reach:</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```python
sum_win_prob = 0;
sum_lose_prob = 0;
for( i = each of his hands )
    sum_lose_prob += r[i]
```
### 3: Fast Terminal Node Evaluation

- **Reach:**
  - 0.05
  - 0.1
  - 0.1
  - 0.05
  - 0.1
  - 0.1

- **Value:**

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Reach</td>
<td>0.05</td>
<td>0.1</td>
<td>0.1</td>
<td>0.05</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**sum_win_prob = 0 \quad sum_lose_prob = 0.5**

```c
for( s = each set of equal-strength hands )
    for( i = each tied hand in s )
        sum_lose_prob -= r[i];
    for( i = each tied hand in s )
        v[i] = -u*sum_lose_prob + u*sum_win_prob
    for( i = each tied hand in s )
        sum_win_prob += r[i];
```

Home

Wednesday, November 14, 2012
3: Fast Terminal Node Evaluation

for( s = each set of equal-strength hands )
    for( i = each tied hand in s )
        sum_lose_prob -= r[i];
    for( i = each tied hand in s )
        v[i] = -u*sum_lose_prob + u*sum_win_prob
    for( i = each tied hand in s )
        sum_win_prob += r[i];

Reach: 0.05 0.1 0.1 0.05 0.1 0.1
Value:

sum_win_prob = 0  sum_lose_prob = 0.5
3: Fast Terminal Node Evaluation

for( s = each set of equal-strength hands )
    for( i = each tied hand in s )
        sum_lose_prob -= r[i];
    for( i = each tied hand in s )
        v[i] = -u*sum_lose_prob + u*sum_win_prob
    for( i = each tied hand in s )
        sum_win_prob += r[i];

sum_win_prob = 0          sum_lose_prob = 0.45
for (s = each set of equal-strength hands) 
  for (i = each tied hand in s) 
    sum_lose_prob -= r[i];
for (i = each tied hand in s) 
  v[i] = -u*sum_lose_prob + u*sum_win_prob
for (i = each tied hand in s) 
  sum_win_prob += r[i];

sum_win_prob = 0  sum_lose_prob = 0.35

<table>
<thead>
<tr>
<th>Reach:</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3: Fast Terminal Node Evaluation

Reach: | 0.05 | 0.1 | 0.1 | 0.05 | 0.1 | 0.1 |
Value: | -0.35u | | | | |

\[
\text{sum\_win\_prob} = 0 \quad \text{sum\_lose\_prob} = 0.35
\]

\[
\text{for( s = each set of equal-strength hands )}
\]
\[
\text{for( i = each tied hand in s )}
\]
\[
\text{sum\_lose\_prob} -= r[i];
\]
\[
\text{for( i = each tied hand in s )}
\]
\[
v[i] = -u*\text{sum\_lose\_prob} + u*\text{sum\_win\_prob}
\]
\[
\text{for( i = each tied hand in s )}
\]
\[
\text{sum\_win\_prob} += r[i];
\]
3: Fast Terminal Node Evaluation

for( s = each set of equal-strength hands )
for( i = each tied hand in s )
    sum_lose_prob -= r[i];
for( i = each tied hand in s )
    v[i] = -u*sum_lose_prob + u*sum_win_prob
for( i = each tied hand in s )
    sum_win_prob += r[i];

Reach:
<table>
<thead>
<tr>
<th></th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
</tr>
</thead>
</table>
Value:| -0.35u | -0.35u |

sum_win_prob = 0  sum_lose_prob = 0.35
3: Fast Terminal Node Evaluation

for( s = each set of equal-strength hands )
    for( i = each tied hand in s )
        sum_lose_prob -= r[i];
    for( i = each tied hand in s )
        v[i] = -u*sum_lose_prob + u*sum_win_prob
    for( i = each tied hand in s )
        sum_win_prob += r[i];

Reach:

<table>
<thead>
<tr>
<th>Value</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
</tr>
</thead>
</table>

Value:

<table>
<thead>
<tr>
<th>Value</th>
<th>-0.35u</th>
<th>-0.35u</th>
</tr>
</thead>
</table>

sum_win_prob = 0.05    sum_lose_prob = 0.35
3: Fast Terminal Node Evaluation

for (s = each set of equal-strength hands)
for (i = each tied hand in s)
    sum_lose_prob -= r[i];
for (i = each tied hand in s)
    v[i] = -u*sum_lose_prob + u*sum_win_prob
for (i = each tied hand in s)
    sum_win_prob += r[i];

<table>
<thead>
<tr>
<th>Reach</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>-0.35u</td>
</tr>
<tr>
<td>0.1</td>
<td>-0.35u</td>
</tr>
<tr>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td></td>
</tr>
</tbody>
</table>

Reach: 0.05, 0.1, 0.1, 0.05, 0.1, 0.1
Value: -0.35u, -0.35u

\[ \text{sum\_win\_prob} = 0.15 \quad \text{sum\_lose\_prob} = 0.35 \]
3: Fast Terminal Node Evaluation

for( s = each set of equal-strength hands )
    for( i = each tied hand in s )
        sum_lose_prob -= r[i];
    for( i = each tied hand in s )
        v[i] = -u*sum_lose_prob + u*sum_win_prob
    for( i = each tied hand in s )
        sum_win_prob += r[i];

<table>
<thead>
<tr>
<th>Reach:</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value:</td>
<td>-0.35u</td>
<td>-0.35u</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

sum_win_prob = 0.15    sum_lose_prob = 0.25
3: Fast Terminal Node Evaluation

Reach:
<table>
<thead>
<tr>
<th></th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
</tr>
</thead>
</table>
Value:| -0.35u | -0.35u |      |      |      |      |

\[
\text{sum\_win\_prob} = 0.15 \quad \text{sum\_lose\_prob} = 0.20
\]

for ( s = each set of equal-strength hands )
for ( i = each tied hand in s )
    \[
    \text{sum\_lose\_prob} -= r[i];
    \]
for ( i = each tied hand in s )
    \[
    v[i] = -u*\text{sum\_lose\_prob} + u*\text{sum\_win\_prob}
    \]
for ( i = each tied hand in s )
    \[
    \text{sum\_win\_prob} += r[i];
    \]
### 3: Fast Terminal Node Evaluation

#### Code Snippet:

```c
for( s = each set of equal-strength hands )
    for( i = each tied hand in s )
        sum_lose_prob -= r[i];
    for( i = each tied hand in s )
        v[i] = -u*sum_lose_prob + u*sum_win_prob
    for( i = each tied hand in s )
        sum_win_prob += r[i];
```

#### Table:

<table>
<thead>
<tr>
<th>Reach</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>-0.35u</td>
</tr>
<tr>
<td>0.1</td>
<td>-0.35u</td>
</tr>
<tr>
<td>0.1</td>
<td>0.05u</td>
</tr>
<tr>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td></td>
</tr>
</tbody>
</table>

- `sum_win_prob = 0.15`
- `sum_lose_prob = 0.20`
3: Fast Terminal Node Evaluation

for( s = each set of equal-strength hands )
  for( i = each tied hand in s )
    sum_lose_prob -= r[i];
  for( i = each tied hand in s )
    v[i] = -u*sum_lose_prob + u*sum_win_prob
  for( i = each tied hand in s )
    sum_win_prob += r[i];

Reach:

<table>
<thead>
<tr>
<th></th>
<th>0.05</th>
<th>0.10</th>
<th>0.10</th>
<th>0.05</th>
<th>0.10</th>
<th>0.10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>-0.35u</td>
<td>-0.35u</td>
<td>0.05u</td>
<td>0.05u</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{sum\_win\_prob} = 0.15 \quad \text{sum\_lose\_prob} = 0.20
\]
3: Fast Terminal Node Evaluation

for( s = each set of equal-strength hands )
  for( i = each tied hand in s )
    sum_lose_prob -= r[i];
  for( i = each tied hand in s )
    v[i] = -u*sum_lose_prob + u*sum_win_prob
  for( i = each tied hand in s )
    sum_win_prob += r[i];

<table>
<thead>
<tr>
<th>Reach:</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value:</td>
<td>-0.35u</td>
<td>-0.35u</td>
<td>0.05u</td>
<td>0.05u</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

sum_win_prob = 0.25  sum_lose_prob = 0.20
3: Fast Terminal Node Evaluation

for( s = each set of equal-strength hands )
for( i = each tied hand in s )
    sum_lose_prob -= r[i];
for( i = each tied hand in s )
    v[i] = -u*sum_lose_prob + u*sum_win_prob
for( i = each tied hand in s )
    sum_win_prob += r[i];

<table>
<thead>
<tr>
<th>Reach:</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value:</td>
<td>-0.35u</td>
<td>-0.35u</td>
<td>0.05u</td>
<td>0.05u</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

sum_win_prob = 0.3  sum_lose_prob = 0.20
3: Fast Terminal Node Evaluation

for( s = each set of equal-strength hands )
    for( i = each tied hand in s )
        sum_lose_prob -= r[i];
    for( i = each tied hand in s )
        v[i] = -u*sum_lose_prob + u*sum_win_prob
    for( i = each tied hand in s )
        sum_win_prob += r[i];

Reach:

<table>
<thead>
<tr>
<th></th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>-0.35u</td>
<td>-0.35u</td>
<td>0.05u</td>
<td>0.05u</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

sum_win_prob = 0.3  sum_lose_prob = 0.10

Home

Wednesday, November 14, 2012
3: Fast Terminal Node Evaluation

for (s = each set of equal-strength hands)
    for (i = each tied hand in s)
        sum_lose_prob -= r[i];
    for (i = each tied hand in s)
        v[i] = -u*sum_lose_prob + u*sum_win_prob
    for (i = each tied hand in s)
        sum_win_prob += r[i];

<table>
<thead>
<tr>
<th>Reach:</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value:</td>
<td>-0.35u</td>
<td>-0.35u</td>
<td>0.05u</td>
<td>0.05u</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sum_win_prob</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sum_lose_prob</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Wednesday, November 14, 2012
3: Fast Terminal Node Evaluation

for( s = each set of equal-strength hands )
for( i = each tied hand in s )
    sum_lose_prob -= r[i];
for( i = each tied hand in s )
    v[i] = -u*sum_lose_prob + u*sum_win_prob
for( i = each tied hand in s )
    sum_win_prob += r[i];

Reach:

<table>
<thead>
<tr>
<th></th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
<th>0.05</th>
<th>0.1</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>-0.35u</td>
<td>-0.35u</td>
<td>0.05u</td>
<td>0.05u</td>
<td>0.3u</td>
<td></td>
</tr>
</tbody>
</table>

\text{sum\_win\_prob} = 0.3 \quad \text{sum\_lose\_prob} = 0.0
3: Fast Terminal Node Evaluation

Reach: 0.05  0.1  0.1  0.05  0.1  0.1
Value: -0.35u  -0.35u  0.05u  0.05u  0.3u  0.3u

\[ \text{sum}_\text{win}_\text{prob} = 0.3 \quad \text{sum}_\text{lose}_\text{prob} = 0.0 \]

for( s = each set of equal-strength hands )
  for( i = each tied hand in s )
    \text{sum}_\text{lose}_\text{prob} -= r[i];
  for( i = each tied hand in s )
    v[i] = -u*sum_lose_prob + u*sum_win_prob
  for( i = each tied hand in s )
    \text{sum}_\text{win}_\text{prob} += r[i];